Introducción

El reproductor de música es una aplicación web que utiliza HTML, CSS y JavaScript para ofrecer una experiencia de usuario intuitiva y atractiva para reproducir archivos de audio. La aplicación está construida siguiendo el patrón de diseño Model-View-Controller (MVC), lo que permite separar la lógica de negocio de la interfaz de usuario y los datos.

El modelo maneja los datos del reproductor, como la lista de canciones, el estado de la canción actual y la duración total del archivo de audio. La vista se encarga de presentar los datos al usuario en la pantalla, como la información de la canción actual, la barra de progreso y los controles de reproducción. El controlador actúa como intermediario entre el modelo y la vista, y maneja las interacciones del usuario con la aplicación.

El reproductor de música utiliza el elemento de audio HTML5 para reproducir archivos de audio. La aplicación también cuenta con funciones para cargar la lista de canciones, cambiar entre canciones, pausar y reanudar la reproducción, ajustar el volumen y actualizar la barra de progreso de la canción actual.

Además, el reproductor de música utiliza eventos de JavaScript para detectar la interacción del usuario y actualizar dinámicamente la interfaz de usuario en respuesta. La aplicación también utiliza técnicas de manipulación del DOM para actualizar los elementos HTML en la página en tiempo real.

```
// Cargar y mostrar lista de canciones
function loadSongs() {
  songsList.map((song, index) => {
    const li = document.createElement("li");
    const link = document.createElement("a");
    link.textContent = song.Title;
    link.href = "#";
    link.addEventListener("click", () => loadSong(index));
    link.addEventListener(
      "mouseover",
      () => (link.style.textDecoration = "underline")
    );
    link.addEventListener(
      "mouseout",
      () => (link.style.textDecoration = "none")
    );
    li.appendChild(link);
    songs.appendChild(li);
    audio.setAttribute("value2", "primero-lista");
  });
}
```

En este fragmento se utiliza un array llamado "songsList" que contiene objetos que representan canciones con sus títulos, archivos de audio y carátulas. Además, se define una función llamada "loadSongs" que crea una lista de elementos de enlace para cada canción y los agrega a la lista de canciones del reproductor. Finalmente, se llama a la función "loadSongs" para cargar y mostrar la lista de canciones en el reproductor.

```
//Movimiento de barra de progreso
function progress() {
    let currentTimePorcentual = (audio.currentTime * 100) / audio.duration;
    prog.style.width = currentTimePorcentual + "%";
    let currentTimeSeg = audio.currentTime.toFixed(0);
    let duration = audio.duration.toFixed(0);
    durationIndicator(duration);
    timeIndicator(currentTimeSeg);
    if (audio.ended) {
      stopSong();
    }
  }
```

Este fragmento de código define una función llamada "progress" que se encarga de

actualizar la barra de progreso del reproductor de música. Primero, se calcula el porcentaje del tiempo actual de la canción que se ha reproducido en relación con su duración total y se actualiza el ancho de la barra de progreso en consecuencia. Luego, se establecen dos indicadores para mostrar el tiempo actual y la duración total de la canción, y se verifica si la canción ha finalizado. Si es así, se llama a la función "stopSong" para detener la canción.

```
//Cambiando color de fondo

function getAverageColor(imageElement, ratio) {
  const canvas = document.createElement("canvas");

  let height = (canvas.height = imageElement.naturalHeight);
  let width = (canvas.width = imageElement.naturalWidth);

  const context = canvas.getContext("2d");
  context.drawImage(imageElement, 0, 0);

  let data, length;
  let i = -4,
    count = 0;

  try {
    data = context.getImageData(0, 0, width, height);
    length = data.data.length;
  } catch (err) {
    console.error(err);
    return {
      R: 0,
      G: 0,
      B: 0,
    };
  }
  let R, G, B;
  R = G = B = 0;

  while ((i += ratio * 4) < length) {
    ++count;

    R += data.data[i];
    G += data.data[i + 1];
    B += data.data[i + 2];
  }

  R = ~~(R / count);
  G = ~~(G / count);
```

```
  B = ~~(B / count);

  return {
    R,
    G,
    B,
  };
}

const image = document.querySelector("img");
image.onload = () => {
  const { R, G, B } = getAverageColor(image, 4);
```

Este código usa la función getAverageColor() para extraer el color dominante de un elemento de imagen y luego aplica el color a varios elementos de la interfaz del reproductor.

La función getAverageColor() toma dos argumentos: un elemento de imagen y una proporción (que determina cuántos píxeles deben omitirse al muestrear los colores de la imagen). Crea un elemento canvas y dibuja la imagen en él. Luego, recorre los datos de píxeles del canvas y calcula el valor RGB promedio de los píxeles en función de la proporción dada. Finalmente, devuelve un objeto que contiene los valores R, G y B.

Una vez cargada la imagen, se activa el evento de carga y se llama a la función getAverageColor() con el elemento de imagen y una proporción de 4. Los valores RGB resultantes se utilizan para generar degradados para los elementos playerColor, colorProg y volSliderColor, como así como un color de fondo sólido para los elementos colorProgCont y SlidContColor.

En general, este código cambia dinámicamente el esquema de color de la interfaz del reproductor según los colores de la imagen cargada.

```
//Cargar canción seleccionada
function loadSong(songIndex) {
    audio.src = "/musica/" +
songsList[songIndex].File;
    audio.play();
    LoadCoverSong(songIndex);
    LoadTitleSong(songIndex);
    changeActiveClass(songIndex);
    next.addEventListener("click", () =>
loadNextSong(songIndex));
    prev.addEventListener("click", () =>
loadPrevSong(songIndex));
    playPauseIcon.className = "fas fa-pause";
    setProgressCont.addEventListener("mousemove",
(event) => showTime(event));
    audio.addEventListener("ended", () => {
      stopSong();
      loadSong(songIndex + 1);
    });

    audio.setAttribute("value2", songIndex);
    durationI.className = "audio-duration";
    currentT.className = "progress-seg";
    timeInfo.className = "time-info";
    actualSong=songIndex;
  }
```

Este código carga una canción seleccionada en un reproductor de música.

La función loadSong recibe un parámetro songIndex, que es un número que indica la posición de la canción seleccionada en una lista de canciones llamada songsList.

Dentro de la función, se establece la fuente de audio del reproductor a través de la propiedad src del elemento audio. Luego se reproduce la canción con el método play(). Además, se cargan la portada de la canción y el título de la canción correspondiente mediante las funciones LoadCoverSong y LoadTitleSong. También se cambia la clase de un elemento activo mediante la función changeActiveClass.

Se establece un evento click en los botones "siguiente" y "anterior" para cargar la siguiente o anterior canción en la lista de canciones, respectivamente. El icono del botón de reproducción/pausa se cambia a "pausa" con la clase "fas fa-pause".

También se establece un evento mousemove en un elemento contenedor de la barra de progreso para mostrar el tiempo de la canción en el que se encuentra el puntero del mouse.

Finalmente, se establece un evento ended en el elemento audio para que cuando la canción termine de reproducirse, se cargue la siguiente canción en la lista de canciones. El índice de la canción seleccionada se establece en un atributo value2 del elemento audio.

La función devuelve el índice de la canción seleccionada almacenado en la variable actualSong.

```
//Mostrar texto emergente del tiempo al poner el puntero sobre la barra
function showTime(event) {
    let barPosition = ((event.clientX - 62) * 100) / 340;
    const barRed = barPosition.toFixed(0);
    let currentTime = ((barRed * audio.duration) / 100).toFixed(0);

    let x = event.clientX;
    let y = event.clientY;

    if (
      event.target.className === "progress-container" ||
      event.target.className === "progress"
    ) {
      timeInfo.style.left = x + 10 + "px";
      timeInfo.style.top = y + 10 + "px";
    }

    let min = Math.floor((currentTime / 60) % 60);
    min = min < 10 ? "0" + min : min;
    let sec = currentTime % 60;
    sec = sec < 10 ? "0" + sec : sec;
    timeInfo.innerHTML = min + ":" + sec;

    if (barRed >= 0 && barRed <= 100) {
      timeInfo.style.visivility = "visible";
    }
  }
```

La función showTime(event) muestra un texto emergente con el tiempo correspondiente a la posición actual del puntero sobre la barra de progreso de la canción. Para ello, se obtiene la posición actual del puntero en la barra y se calcula el tiempo correspondiente según la duración total de la canción.

Luego se establecen las coordenadas de la posición del texto emergente cerca del puntero. Si el puntero está sobre la barra, el texto emergente se hace visible y se muestra el tiempo actualizado en el formato "mm:ss".

```
/Al hacer click
function setProgress(event) {
  let barPosition = ((event.clientX - 62) * 100) / 340;
    const barRed = barPosition.toFixed(0);
    let currentTimeSegSet = (barRed * audio.duration) / 100;
    audio.currentTime = currentTimeSegSet;
  }

  //Al mantener presionado y mover el mouse
  function presion(event) {
    if (setProgressCont.getAttribute("value") == "true") {
      setProgress(event);
    }
  }

  function click(event) {
    let ev = event.isTrusted;
    setProgressCont.setAttribute("value", ev);
  }
  function clickUp(event) {
    if (
      event.target.className === "progress-container" ||
      event.target.className === "progress" ||
      event.target.className !== "progress-container" ||
      event.target.className !== "progress"
    ) {
      setProgressCont.setAttribute("value", "false");
    }
  }
```

Estas funciones manejan la interacción del usuario con la barra de progreso del reproductor de audio. Cuando se hace clic en la barra de progreso, setProgress() se activa para establecer el tiempo actual del audio en función de la posición de la barra de progreso en la que se hizo clic. Cuando se mantiene presionado el mouse y se mueve, presion() se activa para establecer el tiempo actual del audio mientras se mueve la barra de progreso. Cuando se hace clic en la barra de progreso para presionar el botón del mouse, click() se activa para establecer el valor del atributo value del elemento progress-container. Cuando se suelta el botón del mouse, clickUp() se activa para establecer el valor del atributo value en "false". Estos atributos se utilizan para determinar si se debe seguir actualizando el tiempo actual del audio mientras se mueve la barra de progreso o no.

```
//Funciones de volumen
audio.volume = 0.5;
let x;
function volume(event) {
  console.log(event.pageX);
  if (event.pageX >= 68 && event.pageX <= 216) {
    const sliderPorcentual = (((event.pageX - 68) * 100) / 147).toFixed(0);
    const volumePorcentual = sliderPorcentual + "%";
    volumeSlider.style.width = volumePorcentual;
    audio.volume = sliderPorcentual / 100;
    x = sliderPorcentual;
    if (sliderPorcentual <= 35){
      volumeIcon.className= "fas fa-volume-down"
    }else if(sliderPorcentual >= 36){
      volumeIcon.className= "fas fa-volume-up"
    }
  }
}

function presionVol(event) {
  if (sliderCont.getAttribute("value") == "true") {
    volume(event);
  }
}

function clickVol(event) {
  let ev = event.isTrusted;
  sliderCont.setAttribute("value", ev);
}
function clickUpVol(event) {
  if (
    event.target.className === "sliderCont" ||
    event.target.className === "volumeSlider" ||
    event.target.className !== "sliderCont" ||
    event.target.className !== "volumeSlider"
  ) {
    sliderCont.setAttribute("value", "false");
  }
}

function mute() {
  if (audio.volume !== 0) {
    if (audio.volume === 0.5) {
      x = 50;
    }
    audio.volume = 0;
    volumeIcon.className = "fas fa-volume-mute";
  } else if (audio.volume === 0) {
```

```
    const sliderPorcentual = x;

    audio.volume = sliderPorcentual / 100;
    x = sliderPorcentual;
    if (sliderPorcentual <= 35){
      volumeIcon.className= "fas fa-volume-down"
    }else if(sliderPorcentual >= 36){
      volumeIcon.className= "fas fa-volume-up"
    }
  }
}
```

Este código controla el volumen de un reproductor de audio. Crea un elemento deslizante que se puede mover para ajustar el volumen y también incluye un botón de silencio.

El código primero define algunas variables, incluido el elemento volumeSlider, que es la barra que el usuario mueve para ajustar el volumen, y el elemento volumeContainer, que contiene el control deslizante y aparece cuando el usuario pasa el cursor sobre el ícono de volumen.

Luego define varios oyentes de eventos, incluido uno que cambia la visibilidad del contenedor de volumen y otro que silencia el audio cuando se hace clic en el botón de silencio. También hay un detector de eventos para cuando el usuario hace clic en el control deslizante de volumen, que llama a la función de volumen para ajustar el volumen según la posición del control deslizante.

La función de volumen toma la posición del clic del usuario y calcula el porcentaje del control deslizante que se ha llenado en función de esa posición. Luego establece el volumen del audio en ese porcentaje y actualiza el ancho del control deslizante en consecuencia. Si el volumen se establece en cero, cambia el icono del botón de silencio para reflejar eso.

También hay varias otras funciones que controlan el comportamiento del control deslizante, incluido presionVol, que ajusta el volumen cuando el usuario mueve el control deslizante mientras mantiene presionado el botón del mouse, y clickVol y clickUpVol, que rastrean si el botón del mouse se mantiene presionado actualmente. . Finalmente, la función de silencio alterna el volumen entre cero y su valor anterior y actualiza el icono del botón de silencio en consecuencia.